ICS 35.240.01
CCS M30 49

Social Organization Standard

T/GBA 015—2024

Technical requirements for enterprise-level low-code development platform

企业级低代码开发平台技术要求

(English Translation)

Issue date: 2024-02-20 Implementation date: 2024-03-20

Issued by Guangdong-Hong Kong-Macao Greater Bay Area Standards Innovation Alliance

Content

Foreword

This stabdard is drafted in accordance with the rules set forth in GB/T 1.1-2020 *Directives for Standardization -- Part 1: Rules for the Structure and Drafting of Standardizing Documents*.

This standard is proposed by Tencent Cloud Computing (Beijing) Co., Ltd.

This standard is prepared by the Industrial Internet Committee of the Guangdong-Hong Kong-Macao Greater Bay Area Standards Innovation Alliance.

This standard is authorized for use by organizational partners and all members of the Guangdong-Hong Kong-Macao Greater Bay Area Standards Innovation Alliance. These members are required to adopt the same transformation into their own group standards and publicize the standard basic information on the National Group Standards Information Platform.

Please note that some of the contents of this document may involve patent rights. The issuing body of this document shall not be held responsible for identifying any or all such patent rights.

This is the first release.

Enterprise-level Low-Code Development Platform Technical Requirements

1 Scope

This document specifies the system architecture and general technical requirements of the enterprise-level low-code development platform, including the domain-specific language (DSL) layer, the core capability layer, the integrated development, the running mode, and the operation and maintenance service layer, etc.

This document is applicable to the design, development and implementation of the enterprise-level low-code development platform.

2 Normative Reference Documents

This document does not have normative reference documents.

3 Terms and Definitions

The following terms and definitions apply to this document.

3.1 Enterprise-level Low-code Development Platform

A platform that provides the capability of developing services/products with reduced or no coding for the digitalization of organizational business.

3.2 Domain Specified Language

A computer programming language with limited expressiveness for a specific domain, which provides the domain development language for the low-code platform.

Note: DSL has the following three characteristics:

a) Language nature: DSL is a programming language, so it must have coherent expressive power, whether it is an expression or a combination of expressions;

b) Limited expressiveness: General-purpose programming languages provide a wide range of capabilities: support for various data, control, and abstract structures. These capabilities are useful, but they also make the language difficult to learn and use. DSL only supports the minimum set of features required by a specific domain. Using DSL, it is impossible to build a complete system, but it can solve some aspects of the system problem;

c) Domain focus: Only in a clear and small domain, this limited-capability language will be useful.

3.3 Logic Visualization

A way or tool to express logic code as logic nodes and lines through a graphical visualization interface, and to arrange code logic, business process and data processing logic, etc.

3.4 UI Visualization

A way to complete the editing and development of pages, layouts, components, containers, plugins, events and templates in UI through interactions including but not limited to component dragging, form filling, etc.

Note 1: Components are the smallest granularity of page layout, and containers can nest components and containers.

Note 2: Events are the basic units of page and component interaction logic.

Note 3: Templates are html pages based on different business logic and technology stacks, which are the carrier units of components and containers.

Note 4: Plugins refer to components without UI.

3.5 Component

The minimum unit required for UI visualization and logical visualization orchestration is collectively referred to as a component, and by orchestrating the components, a complete business logic can be built.

Note: Common types of components include UI components, event components, logic components, template components, etc. Usually, components can be customized and extended by business parties.

3.6 Data Schema

A metadata abstraction collection that describes the data model.

Note: Relevant connection parameters for backend database connection (host, port, etc.).

3.7 Data Model

A model that uses the database and metadata configuration information to automatically generate common data operation interfaces and record these configurations with data schema.

3.8 Write Once Run Multiterminal

A way to generate applications that run on multiple scenarios/platforms with consistent presentation content and interaction logic through a low-code design and development data (including logic, pages, etc.).

3.9 Parsing Engine

An engine that takes the data of the UI schema and the logic schema of the interface visualization as input, and parses the running program framework through runtime.

3.10 Compiling Engine

Translate a program framework into a specified high-level language by applying certain rules to the processing of UI schema and Logic schema.

3.11 Grayscale Publishing

A publishing method that supports smooth transition, supports selecting a part of users according to a certain strategy, and lets them experience the new version of the product function first, and then decides to continue to enlarge the new version of the delivery scope until full upgrade or rollback to the old version by collecting the feedback of these users on the new version function and monitoring the service running status, and analyzing on the new version function, performance, stability and other indicators.

3.12 Hot Fix

A way to change the original code logic or resource file of the App or SDK without re-downloading and installing by dynamically issuing and loading code.

Note: The hot fix of the low-code platform is essentially an update of the schema information. In the dynamic parsing (parsing execution) mode, developers can use configuration and other methods to configure, manage and dynamically issue the schema information, and complete the hot fix capability.

3.13 Low-code Development Platform Product

The application configuration and code data produced by the developer through the low-code development platform, including but not limited to configuration files, data, code, binary files, etc.

4 Abbreviations

The following abbreviations apply to this document.
AI: Artificial Intelligence
API: Application Programming Interface
APP: Application
CPU: Central Processing Unit
DSL: Domain Specified Language
IP: Internet Protocol
JSON: JavaScript Object Notation
OS: Operating System
PC: Personal Computer
PV: Page View
SaaS: Software as a Service
SDK: Software Development Kit
SLA: Service Level Agreement
TCC: Type Correctness Condition
UI: User Interface
XML: eXtensible Markup Language
YAML: YAML Aint a Markup Language

5 System Architecture

Enterprise-level low code development platform, including domain specific language (DSL), core capabilities, integrated development, runtime, operation and maintenance services modules, specifically:

a) Domain specific language module, providing development language capabilities for low code development platforms;
b) Core competency module, providing a core competency engine for low code development platforms, including front-end and back-end logical orchestration, front-end page orchestration and multi end adaptation, back-end data model and data interface orchestration, and other front-end and back-end capabilities;
c) Integrated development, providing the necessary code editing, visual design, AI development capabilities for low code development platforms;
d) The running mode defines the main running mode of low code development platforms;
e) Operations and maintenance services, providing various capability requirements related to low code development platform development products.

The system architecture of the enterprise-level low code development platform is shown in Figure 1.

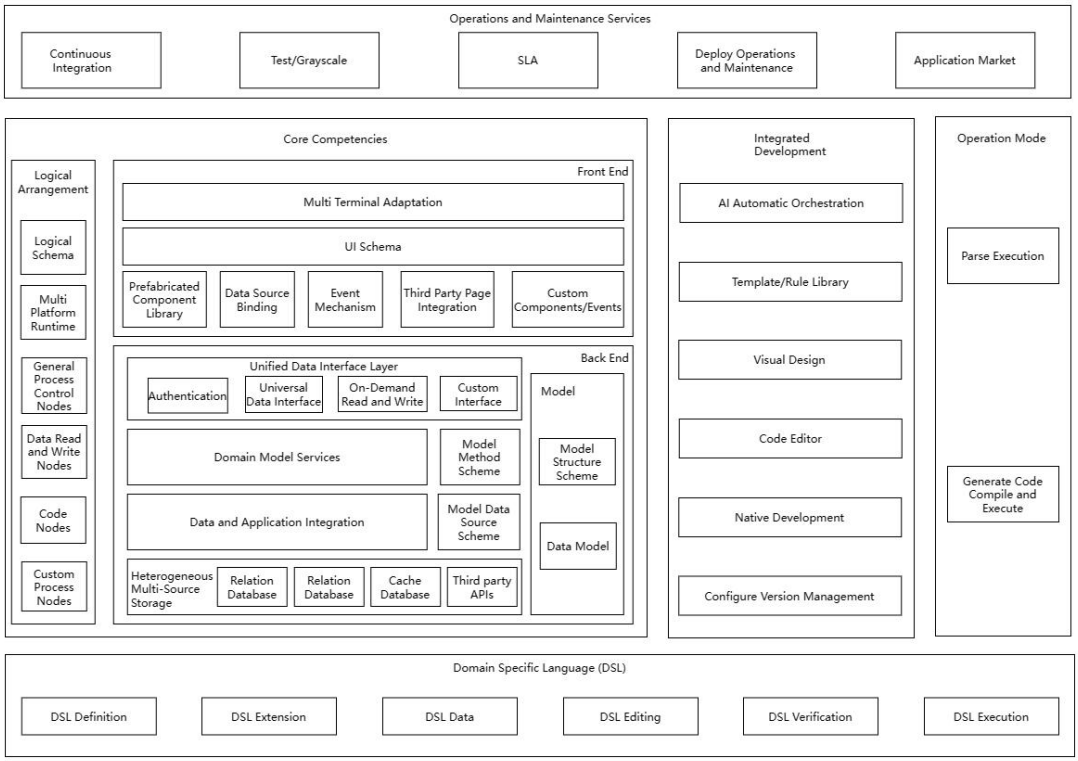

Figure 1 System architecture of the enterprise-level low-code development platform

6 Domain Specified Language (DSL)

Domain specific language (DSL), provides a configuration method for text editing, and meets the following technical requirements:

a) DSL definition: provides the structure of the configuration file required by DSL, including format, syntax, parameters, semantics, domain model, etc.;
b) DSL extension: shall provide the ability of dynamic extension of DSL, to cope with the personalized development needs of business;
c) DSL data: shall provide the configuration data for building and running low-code applications;
d) DSL editing: refers to the editing method of configuration data, shall support manual writing, visual editing or AI automatic generation of three ways;
e) DSL validation: shall have the ability to check and prompt the format and rationality of the DSL produced during DSL editing;
f) DSL execution: shall have the ability to interpret DSL and run it in real time, as well as compile DSL into native code and run it in two ways.

7 Core Capabilities

7.1 Logic Orchestration

The low-code development platform shall provide the capability of logic orchestration, which solves the development capabilities that are not implemented by the built-in components and functional modules of the platform, including but not limited to the following capabilities:

a) Logic schema: Low code platforms shall be able to generate logic schemas that describe the execution process of logic through logical orchestration, including conditions, loops, assignments, calculations, call relationships, etc. The description files can be mainstream format data files such as JSON, YAML, XML;
b) Multi-platform Runtime: It shall support multiple running environments, and have the ability to run the same set of applications on different platforms, achieving cross-platform application development and deployment;
c) General Process Control Nodes: It shall provide the node capability for controlling the business process, realizing functions such as process control, condition judgment, loop, etc., and building various business processes;
d) Data Read and Write Nodes: It shall provide the ability to read and write data in the business process, realizing data persistence and sharing;
e) Custom Process Nodes: It shall provide the nodes to extend the business process, customize the business process, and realize personalized business needs;
f) Code Nodes: It shall provide the ability to write custom code nodes, realize specific business logic, and extend the functionality and flexibility of the low-code platform.

7.2 Front-end

The low-code development platform shall provide the low-code development capability of the user interface, including but not limited to the following requirements:

a) UI Schema:UI schema shall be provided to describe the user interface, describing the metadata collection of the user interface through containers, layouts, components, styles, events, routing, and other content, including but not limited to page layout, styles, sizes, colors, responsiveness to user behavior, and ability to interact with back-end data;
b) Multi-terminal adaptation: It shall have the ability of multi-terminal adaptation, including but not limited to Web, mini-program, android, iOS, windows, mac OS and other terminal types;
c) Pre-built component library: It shall provide the basic components and atomic components required by the business domain, and can extend customized components or capabilities based on the basic capabilities provided by the generator, with flexibility;
d) Data source binding: It shall support the function of binding the data source required in the low-code development process, including but not limited to binding local variables, global variables to the specified database table, interface return or other custom data source capabilities;
e) Event mechanism: It shall have the implementation of event response, including but not limited to event triggers, life cycle event management, and response events of storage, network, UI behaviors;
f) Third-party page integration: It shall have the ability of integrating third-party pages, including but not limited to using page jump, micro-front-end technology;
g) Custom components/events shall meet the following capability requirements:
    1) It support the custom component capability, custom components shall be designed and developed according to the agreed specifications, and access the low-code development platform, for listing and use;

2) It support the custom event capability, custom events allow components to customize event triggers and event responses, and achieve the effect of component linkage by orchestrating custom event triggers and event responses between different components.

7.3 Back-end

The low-code development platform shall provide the model-driven back-end capability, including but not limited to the following requirements:

a) Model:

The model of a low code development platform shall meet the following requirements:

1) Model structure: A specification is provided to define the data model structure, which is used to describe data types, relationships, validation rules, and other data model specifications, including but not limited to JSON, YAML, and XML;
2) Data model: A data model that uses the corresponding model structure is designed for the business system;

b) Domain and model services: Provide a set of services related to the business domain, through which data in the domain can be created, modified and queried;

c) Model methods: Provide a set of operations for model interaction, including create, read, update, delete (CRUD) operations, and have scalability, allowing custom logic;

d) Data and application integration: It shall support the integration of existing data and applications, to achieve the modernization of traditional application systems, connect enterprise data, avoid data islands and chimney applications;

e) Model data source: Define how the data model connects to its data source. Including relational databases, NoSQL databases, file systems or cloud services. It shall provide a configuration method that makes it easy to switch or integrate multiple data sources;

f) The data interface shall meet the following requirements:

1) Authentication: It ensures that the data interaction process must go through a secure authentication step, to verify and authorize data access;
2) General data interface: The front-end application requests data through a unified way;
3) On-demand read and write: Support flexible data access, allowing to read or update specific parts of data as needed;
4) Custom interface: Allow developers to create application-specific interfaces, to support complex or non-standard data interaction requirements;

g) Heterogeneous multi-source storage shall have the ability to access storage resources, including but not limited to:

1) Storage resources shall include but are not limited to relational databases, in memory databases, object storage, file storage, etc., as well as access capabilities to storage resources and middleware such as Redis and big data platforms;
2) It shall support relational databases, for processing data with fixed schema;
3) It should support semi-structured data storage, such as JSON or XML documents, allowing flexible data models and fast data access;
4) It should support cache databases, providing fast data read and write, usually used to improve application performance;
5) It shall be able to integrate and manage data from third-party APIs, to extend the system's functionality and access external data.

8 Integrated Development

8.1 AI automatic orchestration capability

Low code development platforms can provide AI based intelligent programming capabilities, including the ability to automatically generate code in areas such as intelligent generation, design draft conversion, and intelligent building assistance, specifically:

a) Intelligent generation: Based on AI, convert users' natural language descriptions into product pages, product logic, data models, business processes, etc.;
b) Intelligent Building: Based on AI, providing intelligent assistance, automatically recommending relevant industry attributes, scene components, logical nodes, interfaces, etc., improving low code programming efficiency and low code development quality;
c) Design draft conversion page: Support developers to automatically generate pages by uploading design drafts, parse components in the design drafts, automatically convert them to components in low code platforms, and assemble them into pages.

8.2 Template/rule library

Templates and rule libraries shall be provided to improve the reuse rate of components/functions and accelerate the application construction process.

8.3 Visual design

Visual designers shall be provided to implement functions including but not limited to data model modeling, user interface construction, and logical visualization orchestration.

8.4 Code editor

Source code management capabilities shall be provided, including but not limited to:

a) The source code export capability, and the editor shall have the ability to export the source code file to the local;
b) The source code editing capability, including source code insertion, online editing, etc.

8.5 Native development

Native development management and operational capabilities shall be provided, including but not limited to:

a) It support developers to develop applications through the traditional development mode, and integrate them into the products of the low-code platform;
b) Provide the ability to call Open API interfaces and register callbacks, and according to the business form and business process, integrate the visual capabilities into the business flow link, and retain the original logic of the business system to the greatest extent.

8.6 Configuration version management

The ability to manage configuration shall be provided, including but not limited to:

a) The ability to manage versions, maintain information for each version, and allow developers to perform version rollback operations and view information for each version;
b) The ability to handle conflicts, in the process of collaborative development with multiple people, shall have the ability to handle version conflicts;
c) Access third-party configuration version management tools.

9 Running Mode

The running mode of the low-code development platform includes two types:

a) Parsing execution: Follow the low-code component description protocol, application description protocol, low-code application development framework protocol and logic protocol, etc., read the corresponding Schema generated by the UI visualization tool and logic visualization tool, and complete the front-end display and back-end service parsing and running;
b) Enerate code compilation execution: According to the metadata and data interface information contained in the Schema, provide the ability to translate multiple Schema in the platform into high-level languages, and have at least one programming language capability, including but not limited to Java, Nodejs, Golang programming languages.

10 Operations and Maintenance Services

10.1 Continuous Integration

It shall have the ability for continuous integration, including but not limited to:
a) The operating platform provides an advanced language runtime environment and microservices governance framework, supporting heterogeneous language execution;
b) It shall offer load protection features for the entire system, such as entrance traffic control, protection of single computing unit hardware resources (CPU/memory/disk/network card/queues, etc.), and circuit breaker protection for critical paths;
c) It support the real-time or offline reconciliation capability for key system metrics;
d) It support dynamic business scaling, online rollback, and feature degradation.

10.2 Debugging and Testing

It shall have the ability to debug and test during the development process, including but not limited to:
a) Support unit testing, interface testing, stress testing, and security testing,, with the ability to integrate chaos engineering and other capabilities;
b) Support observation and color tracking in grayscale releases, link tracking, and system topology display capabilities.

10.3 Grayscale Release

It shall have the ability for grayscale releases, including but not limited to:
a) Support the selection of specific user groups for new version features in a controllable proportion, with the remaining users using the old version features;
b) For mobile H5 pages and PC page access, support determining the access of front-end resources for new and old versions based on user unique identifiers or source IP;
c) For mini-programs, PC clients, Mac clients, iOS clients, and Android clients, support pushing new version updates based on user unique identifiers;
d) For back-end service access, support determining the access of back-end services for new and old versions based on user unique identifiers or source IP;
e) For back-end microservices, support end-to-end grayscale releases.

10.4 Deployment

It shall have the deployment capability for developing applications, supporting the following related functions:
a) The ability for visual automation and one-click deployment of front-end UI resources and back-end service resources;

b) Have the capability to support the private deployment of exported products;
c) The ability for distributed consistency can be achieved through methods such as distributed transactions and TCC ultimate consistency assurance;
d) Multi-dimensional daily query systems, such as user dimension/request dimension wait;
e) Should provide automated deployment capabilities;
f) Should support deployment parameter settings;
g) Should support containerized deployment of front-end UI resources;
h) Should support containerized deployment of back-end microservices, with support for parallel scalable deployment;
i) Can support serverless deployment of logically edited back-end atomic logic nodes, with support for horizontal scalable deployment;
j) Can support viewing deployment details, visualizing deployment progress, and displaying deployment logs;
k) Can estimate system capacity and proactively adjust deployments in response to business capacity fluctuations.

10.5 Hot Updates

It shall have the requirements for hot update (release) capabilities, including but not limited to:

a) Back-end hot update capability: The ability to smoothly update back-end business logic while ensuring normal service of interfaces;
b) The front-end hot update capability shall meet the following requirements:
   1) H5 page interface, style, and component functionality can be automatically updated during normal application runtime;
   2) For mini-programs, partial logic, UI, and style hot update capabilities;
   3) For Android applications, hot update capabilities for UI, style, and logic;
   4) For iOS clients, limited hot update capabilities according to iOS system requirements.

10.6 Application Monitoring

Low-code development platforms shall have the capability requirements for application monitoring, including but not limited to:

a) Monitor computing resources and related hardware and software resources, such as CPU/memory/network card/hard disk/network, and corresponding cloud devices;
b) Monitor based on system-reported data, using threshold configuration or intelligent detection based on curves;
c) Logic verification for critical services such as storage or qualification;
d) Exception detection and reconciliation for critical system metrics such as traffic and output;
e) Integration with user feedback and other monitoring systems to promptly identify various risks;
f) When alarms occur, provide information on the cause of the exception, scope of impact, relevant personnel, and handling measures;
g) Support toolchain to locate problems, locate problems through data analysis tools, and verify through the log system;
h) Support multi-dimensional monitoring, including but not limited to business resource dimension, system dimension, business dimension, security dimension, user feedback dimension monitoring, etc.

10.7 Security

10.7.1 Authentication Capability

Products developed on low-code platforms shall possess the ability to allocate and manage account permissions, including but not limited to:

a) Account System: The low-code platform shall have the capability to manage the account system, including account system types and account system permissions;

b) User Groups: A user group shall consist of multiple accounts, and the system can maintain multiple user groups simultaneously;

Note: Users refer to the end-users targeted by the low-code development product.

c) Authentication:

When users access the low-code product, it shall have the following authentication capabilities:

1) Authentication Granularity: The system grants permissions to "user/user group" at the granularity of "resource" + "parameter";

2) Authentication Method: Provide authentication interfaces, allowing users to choose whether to enable system authentication when using resources on the low-code platform.

10.7.2 Audit Capability

It shall have the capability for recording and auditing user operations:

a) The ability of recording and storing user operations on metadata, business data, platform data, etc.;

b) Audit precision shall support tenant-level, application-level, object-level, field-level, etc.

10.8 SLA

10.8.1 Platform SLA

The low-code development platform shall provide the SLA of the platform itself to meet the fllowing indicators:

a) Service level indicators include but are not limited to the following indicators:

1) Availability of low-code development platform pages, including the usability of the entire development platform's preview area, editor, form configuration area, material area, menu area, etc., and the correctness and completeness of the displayed content;

2) Availability of related service interfaces of the low-code development platform, including the service availability of all interfaces related to data storage, data processing, engine conversion, etc., across the entire development platform;

b) Service level objectives include but are not limited to the following indicators:

1) Requirements for page access availability;

Note: Calculation method: (Total duration in a certain period - duration when the page is not accessible)/Total duration in a certain period.

2) Requirements for the availability of platform-related service interfaces.

Note: Calculation method: (Total duration in a certain period - duration of interface abnormality)/Total duration in a certain period.

10.8.2 Product SLA

The low-code development platform shall provide the SLA of the developed products to meet the fllowing indicators:

a) Service Level Indicators include but are not limited to the following indicators:

1) Logic Compatibility: Compatibility of front-end logic running on various terminals and hosts (browsers);

2) UI Adaptability: Consistency of front-end UI interfaces and interactive effects across various terminals and hosts (browsers), including the display integrity and interaction consistency on terminals of various sizes;
3) Stability of Interface Access: Service availability of back-end service interface access;

b) Service Level Objectives include but are not limited to the following indicators:
1) Requirements for logic compatibility;
Note: Calculation method: (Total PV of the product in a certain period - PV due to compatibility issues)/Total PV of the product in a certain period.
2) Requirements for UI adaptability;
Note: Calculation method: (Total PV of the product in a certain period - PV due to UI adaptability issues)/Total PV of the product in a certain period.
3) Requirements for the stability of interface access.
Note: Calculation method: (Total number of interface requests for the product in a certain period - number of abnormal interface requests)/Total number of interface requests for the product in a certain period.

10.9 Application Marketplace

The low-code development platform should support publishing to the application marketplace, include but are not limited:

a) Provide mature application modules for developers to quickly use;
b) Developers can submit applications to the application marketplace to enrich the variety of applications available in the market;
c) The application market can support application review function to prevent low-quality applications from entering the market and affecting user experience;
d) The application market can have unified subscription management, such as application purchase payment, price scheme setting, application expiration management, and support for multi tenant SaaS scenarios.

10.10 Internationalization

Low code platform applications should have the ability to configure multiple languages and time zones.

Bibliography

[1] GB/T 18978.151—2014 Human-System Interaction Ergonomics Part 151: Internet User Interface Guidelines
[2] GB/T 24629—2009 XML Schema Markup Rules for Metadata
[3] GB/T 30522—2014 Science and technology infrastructure—Metadata standardization principle and method
[4] GB/T 37729—2019 Information Technology Technical Requirements for Intelligent Mobile Terminal Application Software (APP)